Medi-Caps International School

A

Project Report

ON

BILLING SYSTEM

Session: 2024-25

Submitted to:
Mrs. Neha Neema
PGT

Submitted by:
Darshan Challani (12-A)
Aarna Thakur (12-A)

Department Of Computer Science
Medi-Caps International School
Indore

CERTIFICATE

This is to certify that Master Darshan Challani and Miss Aarna Thakur of class XII, have successfully completed the project on the topic **Billing System** in partial fulfilment of the requirement for the AISSCE Practical Examination of the subject code Informatics Practices (065) Section.

The project work reported here is as per the guidelines of CBSE for AISSCE Practical Examination and it is done under the supervision of the Department of Computer. The project work carried out by him/her is not in the form of any other project work.

Signatures:

Internal Examiner **External Examiner**

School Seal **Principal**

ACKNOWLEDGEMENT

I would like to express my greatest appreciation to all the individuals who have helped and supported me throughout the project. I am grateful to my project guide, Mrs. Neha Neema, for her ongoing support during the project, from initial advice and encouragement, which led to the final report of this project.

I would also like to extend my heartfelt appreciation to my classmates who collaborated with me on this project. Their contributions and collective effort have greatly enriched the outcome.

I express my deep sense of gratitude to respected **Mr. Gopal Agrawal Sir** (Vice Chairman), **Mrs. Gitika Agrawal Ma'am** (Director) & **Mrs. Yogeshwari Rathore** (Principal), Medi-Caps International School for inspiring us to grab & utilize opportunity & ensuring availability of resources required for completion of this project. Their encouragement and belief in our abilities have been a constant source of motivation.

I express my indebtedness to **Mrs. Nidhi Kadam** (Senior Coordinator) for the considerate support & illuminating guidance.

Above all, I am indebted to my parents for their continuous support and encouragement. Their unwavering faith in me and their willingness to lend a helping hand whenever needed have been crucial throughout this journey.

DATE:

Darshan Challani **Aarna Thakur**

TABLE OF CONTENTS

Chapter 1: Introduction

This billing software, developed using **Python** and **MySQL**, provides a streamlined solution for managing billing and inventory operations. The code is divided into two main components: bill.py for generating bills and entry.py for data entry and inventory management. Together, these scripts offer a complete system for handling customer purchases, tracking inventory, and maintaining records efficiently.

1.1 bill.py: Billing and Invoice Generation

This script is responsible for generating invoices based on customer purchases. It interacts with the MySQL database to retrieve item details, calculate the total cost, and update stock quantities.

Key Features:

- **Database Connection**: Establishes a secure connection to the MySQL database (IPPROJECT) to access stock information.
- **Invoice Creation**: Collects customer details (name, mobile number) and retrieves product information using a barcode.
- **Billing Logic**: Calculates total cost, including an 18% tax on the selling price (SRP), and displays the final invoice.
- **Stock Update**: Updates the stock quantity in the database after a successful transaction.
- **Data Export**: Exports billing details to a CSV file for record-keeping.
- **User Interaction**: Allows users to generate another bill or exit the program seamlessly.

1.2 entry.py: Inventory Data Entry and Management

This script facilitates adding new items to the inventory or updating existing stock levels.

Key Features:

- **Database Connection**: Connects to the MySQL database to fetch existing stock data.
- **Barcode Verification**: Checks if an entered barcode already exists in the stock.
- **Item Addition**: Inserts new items into the STOCK table if the barcode is not found.
- **Stock Update**: Updates the quantity of existing items when the barcode matches an existing entry.
- **Error Handling**: Manages database and input errors gracefully, ensuring data integrity.
- **User Interaction**: Allows users to add/update another entry or exit the program.

Technology Stack

- **Python**: Used for the core logic, input handling, and interaction with the MySQL database.
- **MySQL**: Manages inventory data securely and efficiently.

Benefits of the Codebase

- **Efficiency**: Automates billing and inventory updates, reducing manual errors.
- **Data Integrity**: Ensures stock quantities are accurate by updating the database in real-time.
- **User-Friendly Interface**: Simple prompts guide users through billing and data entry processes.
- **Scalability**: Easily adaptable for businesses with varying inventory sizes.

Conclusion

The combination of bill.py and entry.py creates a powerful and efficient billing system that can be customized and expanded as needed. Whether you're generating invoices or managing stock, this system ensures accuracy, security, and ease of use, making it an essential tool for any business.

Chapter 2: Problem Statement & Objective

2.1 **Problem Statement**

Small and medium-sized businesses often face challenges in managing billing and inventory processes efficiently. Manual systems or outdated software frequently led to operational inefficiencies, including inaccurate billing, errors in inventory tracking, and time-consuming record management. These challenges result in reduced productivity, financial losses, and customer dissatisfaction.

Key issues include:

Inefficient Billing Process: Calculating taxes, generating invoices, and tracking payments manually increases the likelihood of errors and delays.

Poor Inventory Management: Businesses struggle to maintain accurate real-time stock levels, leading to overstocking, stockouts, or untraceable discrepancies in inventory.

Data Vulnerability: Without secure and structured data management, customer, product, and transaction records are at risk of being lost, corrupted, or mishandled.

Lack of Integration and Automation: Existing systems often lack the ability to seamlessly update inventory during sales or generate detailed reports, making data analysis and decision-making cumbersome.

User Experience Challenges: Many available systems are overly complex, requiring technical expertise to operate, which poses a barrier for users with minimal technical knowledge.

2.2 Specific Objectives:

There is a pressing need for a robust, automated billing and inventory management system that can streamline these processes, minimize errors, and provide businesses with a user-friendly, secure, and reliable solution.

The primary objective of this project is to **develop a robust, efficient, and user-friendly billing and inventory management system** using **Python** and **MySQL** that automates key business processes.

This includes generating accurate invoices, managing stock levels, and maintaining secure data records to enhance operational efficiency and minimizing manual errors.

1. **Automate Billing Process**:

   Streamline the generation of invoices by automating tax calculations, total cost computation, and payment summaries.

2. **Inventory Management**:

   Enable real-time updates to stock levels during sales and new product entries, ensuring accurate inventory tracking.

3. **Data Security and Integrity**:

   Leverage MySQL for secure, structured storage of customer, product, and transaction data, ensuring data integrity and easy retrieval.

4. **User-Friendly Interface**:

   Provide a simple, interactive interface that guides users through billing and data entry processes, minimizing the need for technical expertise.

5. **Data Export and Reporting**:

Export transaction data to CSV files for record-keeping and generate reports for better business insights and analytics.

6. **Error Handling and Validation**:

   Implement robust input validation and error handling to prevent data inconsistencies and ensure smooth operation.

Outcome:

The software aims to improve the efficiency of billing and inventory processes, reduce manual effort, enhance accuracy, and provide businesses with a reliable tool to manage day-to-day operations seamlessly.

Chapter 3: Concepts Used

3.1 Python:

Python is a high-level, interpreted, and general-purpose programming language known for its simplicity, readability, and versatility. Created by **Guido van Rossum** and first released in **1991**, Python has become one of the most popular languages worldwide due to its ease of use and vast range of applications, from web development to data science.

Key Features of Python

1. **Simple and Readable Syntax**
   Python's syntax is clean and easy to understand, making it an excellent choice for beginners and professionals alike. Its focus on readability allows developers to write clear, concise code.
2. **Interpreted Language**
   Python code is executed line-by-line by an interpreter, which makes debugging easier and allows rapid development.
3. **Dynamically Typed**
   Variables in Python are dynamically typed, meaning you don't need to declare their type explicitly. This flexibility allows faster coding but requires careful error handling.
4. **Extensive Standard Library**
   Python comes with a vast standard library that includes modules for various tasks such as file I/O, regular expressions, web development, and more.
5. **Cross-Platform Compatibility**
   Python is platform-independent, meaning code written on one operating system (like Windows) can run on another (like Linux or macOS) without modification.

6. **Open Source and Community Support**
   Python is open-source and supported by a large, active community. This ensures continuous development and access to countless resources, libraries, and frameworks.

Applications of Python

1. **Web Development**
   Frameworks like **Django** and **Flask** make it easy to build robust web applications quickly.
2. **Data Science and Machine Learning**
   Python is a favorite in data science, with libraries like **Pandas**, **NumPy**, **Matplotlib**, and **Scikit-learn** providing powerful tools for data analysis and machine learning.
3. **Automation and Scripting**
   Python's simplicity makes it ideal for automating repetitive tasks, such as file manipulation, web scraping, and system administration.
4. **Game Development**
   Libraries like **Pygame** enable developers to create simple games with ease.
5. **Scientific Computing**
   Python is used in scientific research for simulations and complex computations, with tools like **SciPy** and **SymPy**.
6. **Artificial Intelligence (AI) and Deep Learning**
   Frameworks like **TensorFlow** and **PyTorch** enable the development of AI and deep learning applications.

Advantages of Python

- **Ease of Learning and Use**: Ideal for beginners due to its intuitive syntax.
- **Versatility**: Supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
- **Large Ecosystem**: Extensive libraries and frameworks for nearly every field.

Conclusion

Python's flexibility, simplicity, and powerful ecosystem make it an ideal language for a wide variety of programming tasks. Whether you're a beginner starting your programming journey or a seasoned developer tackling complex projects, Python provides the tools and resources needed to succeed in today's technology landscape.

3.2 MySQL:

MySQL is an open-source relational database management system (RDBMS) widely used for managing and organizing large amounts of data. It is known for its reliability, scalability, and ease of use, making it a popular choice for web applications, data-driven software, and enterprise solutions. Developed by **Oracle Corporation**, MySQL operates on the Structured Query Language (**SQL**) to perform operations such as querying, updating, and managing databases.

Key Features of MySQL:

1. **Relational Database**:
   MySQL organizes data into tables, where relationships between data are maintained using keys (primary and foreign keys).
2. **Scalability**:
   It supports small applications with minimal data as well as large-scale applications managing terabytes of information.
3. **High Performance**:
   MySQL is optimized for fast query execution, supporting high-performance read and write operations for demanding applications.
4. **Cross-Platform Compatibility**:
   MySQL runs on various operating systems, including Windows, Linux, and macOS, making it versatile for different environments.
5. **Data Security**:
   MySQL ensures data security with robust user authentication, access control, and encryption features.
6. **Replication and Clustering**:
   Supports replication for creating copies of data across multiple servers and clustering for high availability.
7. **Open Source and Community Support**:
   As an open-source project, MySQL is free to use, with a large community providing extensive support and continuous improvements.

Common Uses of MySQL:

- **Web Applications**: Used by platforms like WordPress, Facebook, and Twitter to store and manage user data, content, and transactions.
- **E-commerce**: Tracks products, customers, orders, and payments.
- **Data Warehousing**: Stores large datasets for analysis and reporting.

- **Content Management Systems (CMS)**: Powers CMS platforms like Joomla and Drupal.

Basic Components of MySQL:

1. **Database**: A collection of tables storing related data.
2. **Table**: A structured format that holds data in rows and columns.
3. **Query**: SQL commands used to interact with the database.
4. **Stored Procedures and Triggers**: Scripts that automate database operations and respond to specific events.
5. **Indexes**: Structures that improve the speed of data retrieval.

Advantages of MySQL:

- Free and open source with commercial licensing options.
- Fast and reliable, even under heavy load.
- Comprehensive security features to protect sensitive data.
- Strong integration capabilities with programming languages like Python, PHP, and Java.

Conclusion:

MySQL is a powerful and versatile database management system suitable for applications ranging from simple websites to complex enterprise solutions. Its combination of speed, security, and flexibility makes it a go-to choice for developers and businesses seeking efficient data management solutions.

3.3 CSV:

CSV (Comma-Separated Values) is a widely used file format for storing and exchanging tabular data in a simple, text-based structure. Each line in a CSV file represents a row of data, and each data field is separated by a comma (or another delimiter such as a semicolon or tab).

Key Characteristics of CSV Files:

1. **Simple Structure**:
   CSV files store data in plain text format, making them lightweight and easy to read.
2. **Tabular Format**:
   Data is organized into rows and columns, similar to a spreadsheet, with each line representing a row and commas separating column values.
3. **Platform Independence**:
   CSV files can be opened and edited in a variety of software, including spreadsheet applications (like Microsoft Excel, Google Sheets) and text editors.
4. **No Complex Formatting**:
   Unlike formats like Excel or database files, CSV does not support complex features such as formulas, formatting, or embedded media, making it ideal for simple data storage and transfer.

Benefits of CSV in the Billing Software:

1. **Data Portability**:
   CSV files make it easy to transfer billing records to other systems or share them across different platforms.

2. **Compatibility**:
   They can be opened and processed by numerous applications, ensuring compatibility with various data analysis tools.
3. **Ease of Integration**:
   CSV files can be easily imported into databases, making them ideal for backup and migration purposes.
4. **Human-Readable**:
   Since CSV files are plain text, they are easy to understand and modify manually if necessary.

Use in the Billing Software:

In the **Billing Software**, CSV files are used to:

1. **Export Billing Data**:
   Every generated bill is saved in a CSV file with details such as the customer's name, contact information, items purchased, quantity, price, tax, and total amount.
2. **Maintain Records**:
   CSV files act as a backup for all transactions, providing a simple way to keep historical records of sales.
3. **Facilitate Reporting**:
   The exported CSV files can be imported into tools like Excel or data analysis platforms for further processing and report generation.

Conclusion:

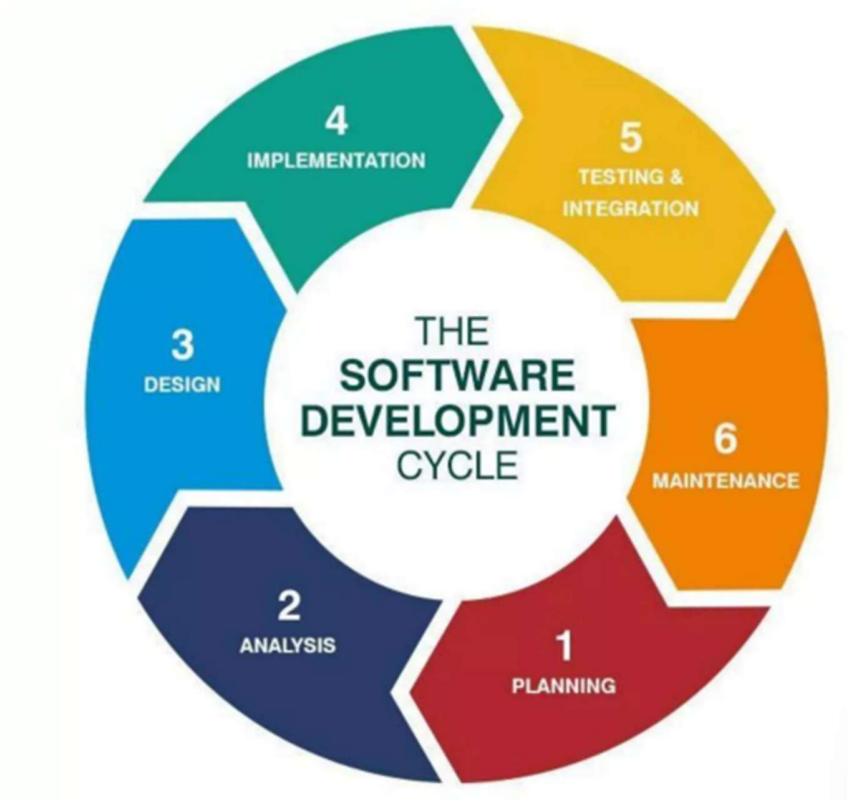
CSV is a powerful and versatile format for data storage and transfer, making it an ideal choice for the **Billing Software** to ensure data portability, simplicity, and broad compatibility.

Chapter 4

Software Development Life Cycle

The software development life cycle is a project management technique that divides complex projects into smaller, more easily managed segments or phases. Segmenting projects allow managers to verify the successful completion of project phases before allocating resources to subsequent phases.

Software development projects typically include initiation, planning, design, development, testing, implementation, and maintenance phases. However, the phase may be divided differently depending on the organization involved.



4.1 Initiation Phase

The initiation phase is the beginning of the project. In this phase, the idea for the project is explored and elaborated. The goal of this phase is to examine the feasibility of the project. In addition, decisions are made concerning who is to carry out the project, which party will be involved and whether the project has an adequate base of support among those who are involved.

In this phase, the current prospective project leader writes a proposal, which contains a description of the above-mentioned matters. Examples of this type of project include business plans and grant applications. The prospective sponsors of the project evaluate the proposal and, upon approval, provide the necessary financing. The project finally begins at the time of approval.

4.2 System Concept Development Phase

The Concept Development Phase may begin after the approval of the Concept Proposal and Project Charter, the completion of the Initiation project status review, and the approval to proceed to the Concept Development Phase.

The focus of the phase is two-fold:

- Evaluate the feasibility of alternatives
- Clearly define and approve project scope, including the system, all deliverables, and all required activities.

The Concept Development Phase activities are inputs into the development of ITPR, which is a required output of this phase.

4.3 Planning Phase

The planning stage (also called feasibility stage) is the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the

objectives for their new systems. By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development. And help to secure the funding and resources they need to make their plan happen.

Perhaps most importantly, the planning stage sets the project schedule, which can be of key importance if development is for a commercial product that must be sent to market at certain times.

4.4 Analysis Phase

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

- Define any prototype system requirements
- Evaluating alternatives to existing prototypes
- Perform research and analysis to determine the needs of end-user

Furthermore, developers will often create software requirement specification or SRS documents.

This includes all the specifications for the software, hardware and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

4.5 Design Phase

The design stage is a necessary precursor to the main developer stage. Developers will first outline the details for the overall application, alongside specific aspects, such as its:

- User interface
- System interface
- Network and network requirements
- Databases

They'll typically turn the SRS document they created into a more logical structure that can later be implemented in a programing language. Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward.

Once complete, development managers will prepare a design document to be referenced throughout the next phase of the SDLC.

4.6 Development Phase

The development stage is the part where developers write code and build the application according to the earlier design documents and outlined specifications.

This is where Static Application Security Testing or SAST tools come into play.

Product program code is built per the design document specifications. In theory, all the prior planning and outlined should make the actual development phase relatively straightforward.

Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters.

Programing languages can include staples such as Python, C++, PHP, and more. Developers will choose the right programing code to use based on the project specifications and requirements.

4.7 Testing Phase

The software must be tested to make sure that there aren't any bugs and that the end-user experience will not negatively be affected at any point. During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed and later retested.

4.8 Implementation and Integration Phase

After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects.

The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

4.9 Maintenance Phase

The SDLC doesn't end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users.

Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

This can include handling residual bugs that were not able to be patched before launch or resolving new issues that crop up due to user reports.

A larger system may require longer maintenance stages compared to smaller systems.

4.10 Disposition Phase

The Disposition Phase is the end of an information system's life cycle. The information system is formally retired according to organizational needs, laws and regulations, and the Disposition Plan. The disposition activities ensure that the information about the system is preserved according to applicable records management regulations and policies for future access.

The decision to proceed with the Disposition Phase is based on recommendations and approvals from an In-process review during the Operations and Maintenance Phase.

Chapter 5: Project SDLC Phases

1. Planning

- **Objective**: Define the scope, requirements, and goals of the project.
- **Activities**:
  - Identify the key functionalities: billing automation, inventory updates, data export, and reporting.
  - Define deliverables such as a secure MySQL database, Python-based scripts, and a user-friendly interface.
  - Allocate resources and create a project timeline.
  - Conduct a feasibility study to ensure the project is practical and aligns with business needs.

2. Requirement Analysis

- **Objective**: Gather and document detailed functional and non-functional requirements.
- **Activities**:
  - Functional Requirements:
    - Automate billing, including tax calculations and payment summaries.
    - Real-time stock updates during sales and inventory entries.
    - Export data to CSV and generate reports.
  - Non-Functional Requirements:
    - The system should handle at least 1,000 transactions per day.
    - Ensure data security with role-based access control.
  - Gather user feedback to understand ease-of-use requirements.
  - Document specifications for database structure (e.g., tables for stock, customers, and transactions).

3. System Design

- **Objective**: Create a blueprint for how the system will function and interact.
- **Activities**:
  - **Database Design**:
    - Design tables for STOCK, CUSTOMERS, and TRANSACTIONS.
    - Establish relationships between tables for efficient data retrieval.
  - **System Architecture**:
    - Choose a client-server architecture: Python as the client and MySQL as the database server.
  - **Workflow Design**:
    - Map out processes such as billing, inventory updates, and data export.
  - **Interface Design**:
    - Design mockups for the user interface using tools like Figma or diagrams for CLI navigation.

4. Implementation

- **Objective**: Develop the software based on the design specifications.
- **Activities**:
  - Set up the MySQL database and implement the required tables.
  - Write Python scripts for:
    - Billing (invoice generation, tax calculations).
    - Inventory management (real-time updates, data validation).
    - Data exports to CSV.
  - Integrate database connectivity using libraries like mysql.connector.
  - Implement error handling and input validation mechanisms.

- Test individual modules during development.

5. Testing

- **Objective**: Ensure the system works as expected and meets all requirements.
- **Activities**:
  - Perform **Unit Testing** for individual scripts (billing, data entry, export).
  - Conduct **Integration Testing** to ensure seamless interaction between Python scripts and the MySQL database.
  - Execute **System Testing** to validate the entire workflow (billing, inventory updates, reporting).
  - Perform **User Acceptance Testing (UAT)** with end-users to confirm usability and functionality.
  - Fix bugs and optimize performance based on feedback.

6. Deployment

- **Objective**: Deliver the system to the end-users and set it up for use.
- **Activities**:
  - Install and configure the system on client machines.
  - Provide documentation for system setup and usage.
  - Train users on system functionality and best practices.
  - Deploy the system in a controlled environment for final feedback.

7. Maintenance

- **Objective**: Ensure the system remains operational, efficient, and updated.

- **Activities**:
  - Monitor system performance and fix any reported bugs.
  - Provide periodic updates to address evolving business needs or security concerns.
  - Enhance features based on user feedback (e.g., add new reporting options or UI improvements).
  - Perform database maintenance, such as backups and optimization.

Deliverables at Each Stage

1. **Planning**: Project timeline, resource allocation, feasibility report.
2. **Requirement Analysis**: SRS (Software Requirements Specification) document.
3. **Design**: Database schema, workflow diagrams, UI mockups.
4. **Implementation**: Source code, database setup scripts.
5. **Testing**: Test cases, bug reports, and UAT feedback.
6. **Deployment**: Installed system, user documentation, and training materials.
7. **Maintenance**: System updates, performance reports.

Chapter 6: Source Code

6.1 home.py

```
print("                    |-------------------------------------------------------------------------|")
print("                    |                        Darshan's Billing Software                       |")
print("                    |-------------------------------------------------------------------------|")
print("                    |                   Press 1 for DATA ENTRY                                |")
print("                    |                   Press 2 for BILLING                                   |")
print("                    |-------------------------------------------------------------------------|")



work=input("                                      Enter here :")

print("                    |-------------------------------------------------------------------------|")
if work=='1':
    print("")
    print("")
    print("")
    import entry
elif work=='2':
    print("")
    print("")
    print("")
    import bill
else:
    print("                        |                  PLEASE ENTER A VALID ENTRY                           |")
    print("                        |-------------------------------------------------------------------------|")
    import home
```

6.2 entry.py (If entered 1)

```
import mysql.connector
import importlib
print("                        |-------------------------------------------------------------------------|")
print("                        |                                DATA ENTRY                               |")
print("                        |-------------------------------------------------------------------------|")
print("                        |                                                                         |")

# Establish database connection
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="useradmin@100",
    database="IPPROJECT"
)

cursor = db.cursor()

# Execute SELECT query to get the existing barcode numbers
cursor.execute("SELECT BARCODE_NUMBER FROM STOCK")

# Fetch all rows
results = cursor.fetchall()

# Create a list of barcode numbers
column_values = [row[0] for row in results]


# Taking input from the user
try:
    barcode_number = int(input("                      |            Enter the BARCODE NUMBER: "))
    name_item = input("                         |            Enter the NAME OF ITEM: ")
    qty = int(input("                       |            Enter the QUANTITY: "))
    mrp = int(input("                       |            Enter the MRP: "))
    srp = int(input("                       |            Enter the SRP: "))

    # Check if barcode_number is in the existing stock
    if barcode_number in column_values:  # Specific check for a barcode
        query="SELECT BARCODE_NUMBER, NAME_OF_ITEM, QUANTITY, MRP, SRP FROM STOCK WHERE BARCODE_NUMBER=%s"
        cursor.execute(query, (barcode_number,))
        #Fetch the first row (if it exists)
        row = cursor.fetchone()
        barcode_number, name_of_item, quantity, mrp, srp = row
        update_query = ("UPDATE STOCK SET QUANTITY = %s "
                        "WHERE BARCODE_NUMBER = %s")
        new_value = quantity + qty  # Adjust this logic if needed
        condition_value = barcode_number
        cursor.execute(update_query, (new_value, condition_value))
        db.commit()  # Commit the update
        print("                        |-------------------------------------------------------------------------|")
        print('                        |                              Item Updated                               |')

    else:
        query = "INSERT INTO STOCK (BARCODE_NUMBER, NAME_OF_ITEM, QUANTITY, MRP, SRP) VALUES (%s, %s, %s, %s, %s);"
        values = (barcode_number, name_item, qty, mrp, srp)
        cursor.execute(query, values)
        db.commit()  # Commit the insert
        print("                        |-------------------------------------------------------------------------|")
        print("                        |                          Item added to stock                            |")
```

```
except mysql.connector.Error as err:
    print(f"                          Database error: {err}")
except ValueError:
    print("                        |Please enter valid numeric values for barcode, quantity, MRP, and SRP.|")
finally:
    cursor.close()
    db.close()
print("                       |-------------------------------------------------------------------------|")
print("")
print("")
print("")
print("                       |-------------------------------------------------------------------------|")
print("                       |                   TO ADD OR UPDATE ANOTHER ENTRY PRESS 1                |")
print("                       |                   PRESS ANY OTHER KEY TO EXIT                           |")
print("                       |-------------------------------------------------------------------------|")
a=input("                                 Enter here :")
print("                       |-------------------------------------------------------------------------|")
if a=='1':
    import entry
    importlib.reload(entry)
else:
    print("                      |                                THANK YOU                                |")
    print("                      |-------------------------------------------------------------------------|")
```

6.3 bill.py

```
import mysql.connector
import importlib
import csv
print("                       |-------------------------------------------------------------------------|")
print("                       |                                MAKE A BILL                              |")
print("                       |-------------------------------------------------------------------------|")
print("                       |                                                                         |")

# Establish database connection
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="useradmin@100",
    database="IPPROJECT"
)

cursor = db.cursor()

name=input("                        |            Enter the Name of Customer :")
number=input("                        |            Enter the Mobile Number :")

# Example SELECT query to fetch a row from the "users" table
barcode_number=int(input("                          |            Enter the Barcode Number :"))
query="SELECT BARCODE_NUMBER, NAME_OF_ITEM, QUANTITY, MRP, SRP FROM STOCK WHERE BARCODE_NUMBER=%s"
cursor.execute(query, (barcode_number,))
# Fetch the first row (if it exists)
row = cursor.fetchone()

# Check if the row is not None (empty result set)
if row:
    # Extract values from the row and assign them to different variables
    barcode_number, name_of_item, quantity, mrp, srp = row

    # Now you can use the variables `user_id`, `username`, `email`, and `age`
    print(f"                      |            BARCODE NUMBER: {barcode_number}                                  |")
    print(f"                      |            NAME OF ITEM: {name_of_item}")
    print(f"                      |            QUANTITY: {quantity}                                          |")
    print(f"                      |            MRP: {mrp}                                                    |")
    print(f"                      |            SRP: {srp}                                                    |")
```

```
else:
    print("                        |            Item Not Found")
    e=input("                         |            Press any key to exit")
    sys.exit()


qty=int(input("                           |            Enter the quantity required :"))

print("                        |-------------------------------------------------------------------------|")
print("")
print("")


print("                        |-------------------------------------------------------------------------|")
print("                        |                                  Bill                                   |")
print("                        |-------------------------------------------------------------------------|")
print("                        |[ Buyer's Name -",name,"] "," "," [ Contact Number -",number,"]|")
print("                        |-------------------------------------------------------------------------|")
print("                                                Name of item -",name_of_item)
print("                                                Quantity -",qty)
print("                                                Price -",srp,'/-')
tax=srp*0.18
print("                                                Tax -",tax,'/-')
total=(srp+tax)*qty
print("                                                Total -",total,'/- ONLY')
print("                        |-------------------------------------------------------------------------|")
update_query = ("UPDATE STOCK SET QUANTITY = %s "
                "WHERE BARCODE_NUMBER = %s")
new_value = quantity - qty  # Adjust this logic if needed
condition_value = barcode_number
cursor.execute(update_query, (new_value, condition_value))
db.commit()  # Commit the update

# Open a file in write mode
with open('output.csv', mode='a', newline='') as file:
    writer = csv.writer(file)

    # Write header (optional)
    writer.writerow(["Name", "Mobile Number", "Item", "Quantity", "Price", "Tax", "Total"])

    # Write data rows
    writer.writerow([name,number,name_of_item,qty,srp,tax,total])

print("                        |                Data exported to CSV file successfully.                  |")
print("                        |-------------------------------------------------------------------------|")
print("")
print("")
print("")

print("                        |-------------------------------------------------------------------------|")
print("                        |                     TO MAKE ANOTHER BILL PRESS 1                        |")
print("                        |                     PRESS ANY OTHER KEY TO EXIT                         |")
print("                        |-------------------------------------------------------------------------|")
a=input("                                  Enter here :")
print("                        |-------------------------------------------------------------------------|")
if a=='1':
    import bill
    importlib.reload(bill)
else:
    print("                        |                                THANK YOU                                |")
    print("                        |-------------------------------------------------------------------------|")
```

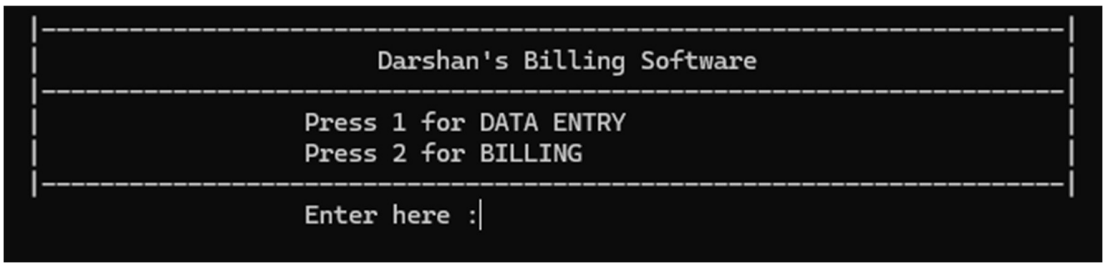
6.4 MySQL Table

```
mysql> SELECT * FROM STOCK;
+----------------+-------------------------------+----------+--------+--------+
| BARCODE_NUMBER | NAME_OF_ITEM                  | QUANTITY | MRP    | SRP    |
+----------------+-------------------------------+----------+--------+--------+
|         100000 | AMAZON ECHO DOT               |      300 |   5999 |   3999 |
|         102048 | ONEPLUS BULLETS WIRELESS Z    |       50 |   5999 |   3999 |
|         134526 | SAMSUNG GALAXY Z FLIP 6       |      452 |  89999 |  83999 |
|         156725 | SAMSUNG A54                   |      183 |  32999 |  28999 |
|         162544 | IPHONE 15 PRO MAX             |      152 | 139999 |  89999 |
|         190765 | IPHONE 16 PRO MAX             |      590 | 144990 | 144900 |
|         192654 | ONEPLUS 10 PRO 5G             |      104 |  59999 |  32999 |
|         234156 | INFINIX ZERO BOOK 13(32GB/1TB)|      250 | 129999 |  69999 |
|         314567 | LENOVO THINKPAD T440S         |      130 |  32000 |  22000 |
|         534267 | SAMSUNG GALAXY NOTE 8         |       40 |  75268 |  12999 |
|         534268 | SAMSUNG GALAXY M32            |      130 |  16999 |   8999 |
|         635246 | IPHONE 13 PRO MAX             |     3873 | 129999 |  56999 |
|         635247 | ONEPLUS 9 5G                  |      235 |  54999 |  21999 |
|         635278 | SAMSUNG GALAXY M31            |      333 |  15999 |   5999 |
|         736251 | ONEPLUS NORD CE2 LITE         |      129 |  21999 |  12999 |
|         762543 | MACBOOK PRO M1                |      124 | 179999 | 148999 |
|         924567 | ONEPLUS 6                     |      130 |  48999 |   8999 |
|         987654 | HUAWEI MATE 20 PRO            |      118 |  69990 |  14999 |
+----------------+-------------------------------+----------+--------+--------+
18 rows in set (0.01 sec)
```

Chapter 7

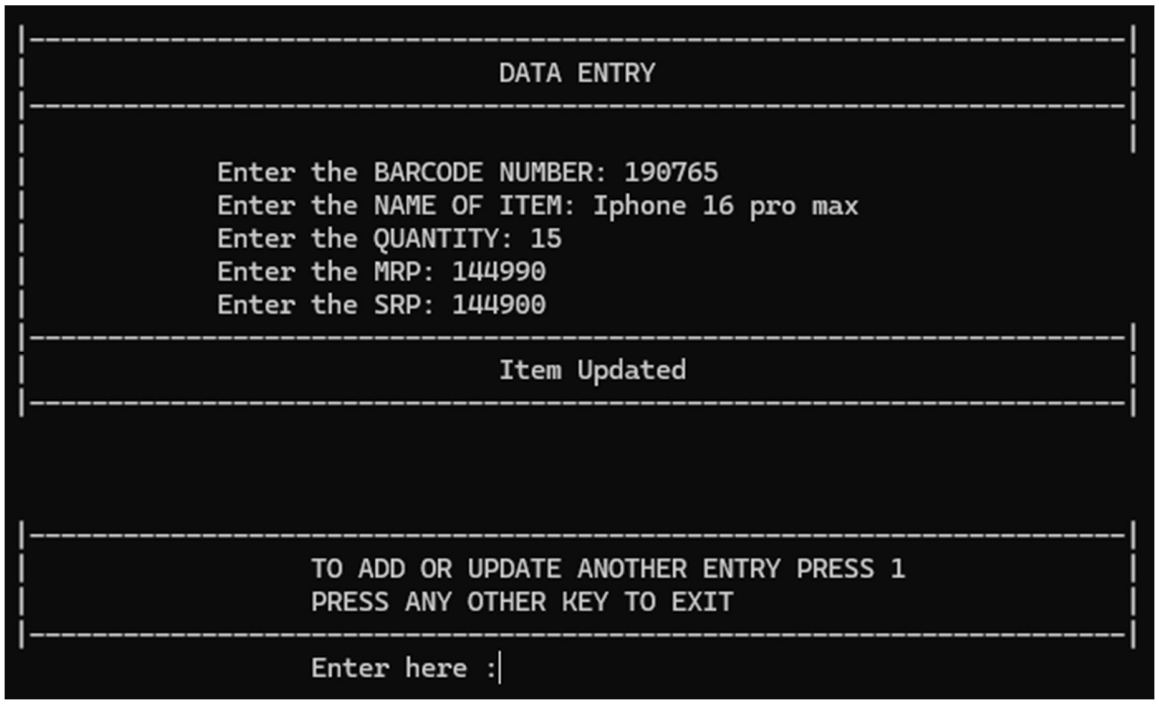
Output

7.1 home.py



7.2 If entered 1 (entry.py)

7.2.1 If already existing barcode entered.



Entry before:

```
|         190765 | IPHONE 16 PRO MAX                |       590 | 144990 | 144900 |
```

Entry after:

```
|         190765 | IPHONE 16 PRO MAX                |       605 | 144990 | 144900 |
```

7.2.2 If a new non-existing barcode entered

```
|-------------------------------------------------------------------------|
|                   TO ADD OR UPDATE ANOTHER ENTRY PRESS 1                |
|                   PRESS ANY OTHER KEY TO EXIT                           |
|-------------------------------------------------------------------------|
                    Enter here :1
|-------------------------------------------------------------------------|
|-------------------------------------------------------------------------|
|                               DATA ENTRY                                |
|-------------------------------------------------------------------------|
|                                                                         |
|             Enter the BARCODE NUMBER: 567098
|             Enter the NAME OF ITEM: FASTRACK REFLEX HELLO
|             Enter the QUANTITY: 120
|             Enter the MRP: 5995
|             Enter the SRP: 3999
|-------------------------------------------------------------------------|
|                           Item added to stock                           |
|-------------------------------------------------------------------------|



|-------------------------------------------------------------------------|
|                   TO ADD OR UPDATE ANOTHER ENTRY PRESS 1                |
|                   PRESS ANY OTHER KEY TO EXIT                           |
|-------------------------------------------------------------------------|
                    Enter here :
```

Table before:

BARCODE_NUMBER	NAME_OF_ITEM	QUANTITY	MRP	SRP
100000	AMAZON ECHO DOT	300	5999	3999
102048	ONEPLUS BULLETS WIRELESS Z	50	5999	3999
134526	SAMSUNG GALAXY Z FLIP 6	452	89999	83999
156725	SAMSUNG A54	183	32999	28999
162544	IPHONE 15 PRO MAX	152	139999	89999
190765	IPHONE 16 PRO MAX	605	144990	144900
192654	ONEPLUS 10 PRO 5G	104	59999	32999
234156	INFINIX ZERO BOOK 13(32GB/1TB)	250	129999	69999
314567	LENOVO THINKPAD T440S	130	32000	22000
534267	SAMSUNG GALAXY NOTE 8	40	75268	12999
534268	SAMSUNG GALAXY M32	130	16999	8999
635246	IPHONE 13 PRO MAX	3873	129999	56999
635247	ONEPLUS 9 5G	235	54999	21999
635278	SAMSUNG GALAXY M31	333	15999	5999
736251	ONEPLUS NORD CE2 LITE	129	21999	12999
762543	MACBOOK PRO M1	124	179999	148999
924567	ONEPLUS 6	130	48999	8999
987654	HUAWEI MATE 20 PRO	118	69990	14999

18 rows in set (0.00 sec)

Table after:

BARCODE_NUMBER	NAME_OF_ITEM	QUANTITY	MRP	SRP
100000	AMAZON ECHO DOT	300	5999	3999
102048	ONEPLUS BULLETS WIRELESS Z	50	5999	3999
134526	SAMSUNG GALAXY Z FLIP 6	452	89999	83999
156725	SAMSUNG A54	183	32999	28999
162544	IPHONE 15 PRO MAX	152	139999	89999
190765	IPHONE 16 PRO MAX	605	144990	144900
192654	ONEPLUS 10 PRO 5G	104	59999	32999
234156	INFINIX ZERO BOOK 13(32GB/1TB)	250	129999	69999
314567	LENOVO THINKPAD T440S	130	32000	22000
534267	SAMSUNG GALAXY NOTE 8	40	75268	12999
534268	SAMSUNG GALAXY M32	130	16999	8999
567098	FASTRACK REFLEX HELLO	120	5995	3999
635246	IPHONE 13 PRO MAX	3873	129999	56999
635247	ONEPLUS 9 5G	235	54999	21999
635278	SAMSUNG GALAXY M31	333	15999	5999
736251	ONEPLUS NORD CE2 LITE	129	21999	12999
762543	MACBOOK PRO M1	124	179999	148999
924567	ONEPLUS 6	130	48999	8999
987654	HUAWEI MATE 20 PRO	118	69990	14999

19 rows in set (0.00 sec)

7.3 If entered 2 (bill.py)

7.3.1 Entries of customer

```
|------------------------------------------------------------------------|
|                              MAKE A BILL                               |
|------------------------------------------------------------------------|
|                                                                        |
|            Enter the Name of Customer :DIVYANSHI VERMA                 |
|            Enter the Mobile Number :7509045855                         |
|            Enter the Barcode Number :190765                            |
|            BARCODE NUMBER: 190765                                      |
|            NAME OF ITEM: IPHONE 16 PRO MAX                             |
|            QUANTITY: 605                                               |
|            MRP: 144990                                                 |
|            SRP: 144900                                                 |
|            Enter the quantity required :2                              |
|------------------------------------------------------------------------|
```

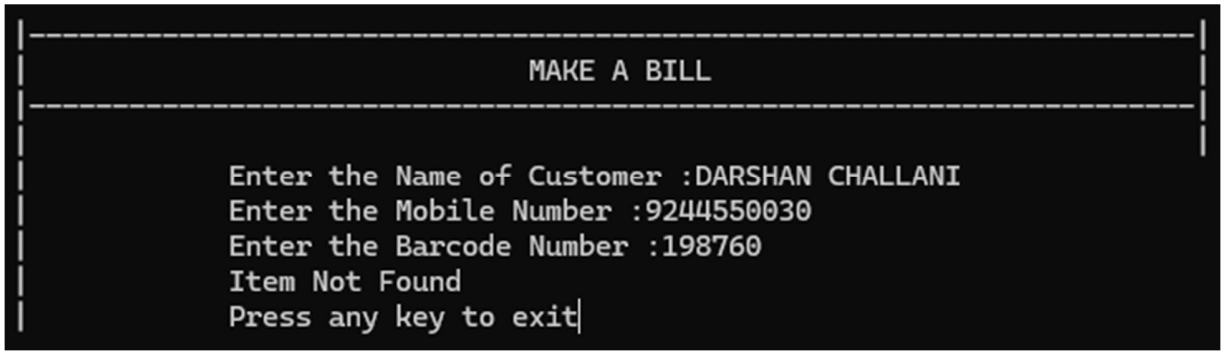
7.3.2 Generated Bill

```
|------------------------------------------------------------------------|
|                                 Bill                                   |
|------------------------------------------------------------------------|
|[ Buyer's Name - DIVYANSHI VERMA ]     [ Contact Number - 7509045855 ]  |
|------------------------------------------------------------------------|
                      Name of item - IPHONE 16 PRO MAX
                      Quantity - 2
                      Price - 144900 /-
                      Tax - 26082.0 /-
                      Total - 341964.0 /- ONLY
|------------------------------------------------------------------------|
|               Data exported to CSV file successfully.                  |
|------------------------------------------------------------------------|



|------------------------------------------------------------------------|
|                   TO MAKE ANOTHER BILL PRESS 1                         |
|                   PRESS ANY OTHER KEY TO EXIT                          |
|------------------------------------------------------------------------|
                    Enter here :
```

7.3.3 CSV Entry of customer

	A	B	C	D	E	F	G
1	Name	Mobile Number	Item	Quantity	Price	Tax	Total
2	Darshan	9244550030	SAMSUNG GALAXY Z FLIP 6	1	83999	15119.82	99118.82
3	Name	Mobile Number	Item	Quantity	Price	Tax	Total
4	Mukes Challani	9300390030	SAMSUNG GALAXY NOTE 8	3	12999	2339.82	46016.46
5	Name	Mobile Number	Item	Quantity	Price	Tax	Total
6	Bharti Challani	9300930030	SAMSUNG GALAXY Z FLIP 6	2	83999	15119.82	198237.64
7	Name	Mobile Number	Item	Quantity	Price	Tax	Total
8	DIVYANSHI VERMA	7509045855	IPHONE 16 PRO MAX	2	144900	26082	341964
9	Name	Mobile Number	Item	Quantity	Price	Tax	Total
10	Bharti	9300390030	SAMSUNG GALAXY Z FLIP 6	1	83999	15119.82	99118.82
11	Name	Mobile Number	Item	Quantity	Price	Tax	Total
12	Darshan	9300930030	SAMSUNG GALAXY Z FLIP 6	1	83999	15119.82	99118.82

7.3.4 If the barcode entered doesn’t exist



Chapter 8: Testing

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics. Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

8.1 Testing Methods

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

8.1.1 Black Box Testing

Black Box Testing is a testing technique where no knowledge of the internal functionality and structure of the system is available. This testing technique treats the system as a black box or closed box. The tester only knows the formal inputs and expected outputs but does not know how the program arrives at those outputs. As a result, all testing must be based on functional specifications. For this reason, black box testing is also considered to be functional testing and is also a form of behavioral testing or opaque box testing or simply closed box testing. Although black box testing is behavioral testing, behavioral test design is slightly different from black box test design because internal knowledge may be available in behavioral testing. It provides an external perspective of the software under test.

Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

8.1.2 Specification Based-Testing

Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Specification-based testing is necessary, but it is insufficient to guard against certain risks.

Advantages of Black Box Testing

- Efficient when used on large systems.
- Since the tester and developer are independent of each other, testing is balanced and unprejudiced.
- Tester can be non-technical.
- There is no need for the tester to have detailed functional knowledge of the system.
- Tests will be done from an end user's point of view, because the end user should accept the system. (This testing technique is sometimes also called Acceptance testing.)
- Testing helps to identify vagueness and contradictions in functional specifications.
- Test cases can be designed as soon as the functional specifications are complete.

Disadvantages of Black Box Testing

- Test cases are challenging to design without having clear functional specifications.
- It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
- It is difficult to identify all possible inputs in limited testing time. As a result, writing test cases may be slow and difficult.
- There are chances of having unidentified paths during the testing process.
- There is a high probability of repeating tests already performed by the programmer.

8.1.3 White-Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) looks inside the software that is being tested and uses that knowledge as part of the testing process. If, for example, exception is thrown under certain conditions, the test might want to reproduce those conditions. White-box testing requires internal knowledge of the system and programming skills. It provides an internal perspective of the software under test.

Types of white box testing:

The following types of white box testing exist:

- API testing - Testing of the application using Public and Private APIs
- Code coverage - creating tests to satisfy some criteria of code coverage.

For example, the test designer can create tests to cause all statements in the program to be executed at least once.

- fault injection methods.
- mutation testing methods.
- static testing - White box testing includes all static testing.

Code Completeness Evaluation

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested

Two common forms of code coverage are:

- Function Coverage: Which reports on functions executed
- Statement Coverage: Which reports on the number of lines executed to complete the test.

They both return coverage metric, measured as a percentage.

Advantages of White-Box Testing

- Testing can commence even before the GUI is ready. A
- As internal functionality is considered, all the possible conditions are considered, and test cases are generated. Hence all the functionalities are being tested.
- It identifies the specific procedure accuracy within the application.
- It minutely verifies whether the program can be successfully executed with other parts of the application.
- It identifies error in the hidden code and thus makes debugging process swift.
- It removes extra lines of code which are not required in the program, thereby optimizing the program and increases the efficiency.
- As the internal coding of the application is considered while preparing test cases, it becomes very easy to identify the input and the expected output data.
- It helps in evaluating all the loops and paths.
- It can provide stability and usability of the test cases.
- Thoroughness achieved in white box testing is far more than black box testing.
- Various hidden defects get unearth while conducting clear box testing.

Disadvantages of White-Box Testing

- As the internal code of the application must be considered while preparing the test cases, skilled testers are required who have knowledge of programming also. Hence the cost of the resources is high.
- It is not possible for the tester to investigate every bit of the code and identify the hidden errors. This may result in failure of the application.
- Sometimes a change in the code may be required and thus all the scenarios may need to be tested again.
- White box testing is an exhaustive method.
- It takes time to test to develop the test cases.
- Test cases are a waste if changes in the implementation code are done frequently.
- If the application is large then complete testing through white box techniques is not feasible.

Chapter 9: Future Work

To ensure scalability and enhance the functionality of the billing and inventory management system, the following improvements and features can be considered for future development:

1. Advanced Reporting and Analytics

Develop detailed dashboards to provide sales insights, stock trends, and Customer purchase patterns. Generate automated daily, weekly, or monthly sales and inventory reports in multiple formats (e.g., PDF). Implement predictive analytics to forecast inventory demand based on historical data.

2. **Payment Gateway Integration:**

Add functionality for processing payments through popular payment gateways.

3. **E-commerce Integration:**

Enable syncing inventory and orders with online platforms such as Amazon, Shopify, or custom e-commerce websites.

4. **Enhanced Security Features**

Implement user authentication and role-based access control (e.g., admin, cashier, stock manager). Add database encryption to protect sensitive data such as customer information and sales transactions.

Chapter 10: Bibliography

- Informatics Practices- Class XI / XII By: Sumita Arora
- Informatics Practices with Python- Class XI By: Preeti Arora
- Informatics Practices - NCERT Textbook for Class XI
- Website: https://www.w3resource.com